## REMARKS

Claims 1-4, 6-15, 17-26 and 28-33 are pending in the present application. By this Response, claims 1, 12 and 23 are amended and claims 5, 16 and 27 are canceled. Claims 1, 12 and 23 are amended to incorporate the subject matter of canceled claims 5, 16 and 27. Reconsideration of the claims in view of the above amendments and the following remarks is respectfully requested.

### I. Application to be Considered Special

This application has received a third non-final Office Action. As per MPEP § 707.02, Applicants respectfully request that the Supervisory Patent Examiner personally check on the pendency of this application and make every effort to terminate prosecution.

### II. 35 U.S.C. § 102, Alleged Anticipation, Claims 1, 12 and 23

The Office Action rejects claims 1-3, 5, 12-14, 16, 23-25 and 27 under 35 U.S.C. § 102(b) as being allegedly anticipated by Zolnowsky (U.S. Patent No. 6,779,182 B1). This rejection is respectfully traversed.

As to claims 1, 12 and 23, the Office Action states:

> As per Claims 1, 12, and 23, Zolnowsky discloses a connection scheduling method, wherein Zolnowsky discloses:
>
> determining if a job is available for scheduling (job scheduling) (at least col. 5, lines 13-21);
>
> determining, in response to said step of determining if said job is available, if a session is available, wherein said session in included in a pool of sessions (threads), said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job (running threads in queue of threads with dispatch priority) (at least col. 6, lines 33-65); and
>
> launching said session to effect said execution of said job, if said session is available (thread (and processor / job) selected for execution) (at least col. 7, lines 17-28; col. 8, lines 43-60).

Office Action dated September 3, 2004, pages 2-3.

Claim 1 is amended to recite subject matter originally presented in claim 5. Claim 1, which is representative of the other rejected independent claims 12 and 23 with regard to similarly recited subject matter, reads as follows:

> 1. A connection scheduling method comprising the steps of:
> determining if a job is available for scheduling;
> determining, in response to said step of determining if said job is available, if a session is available, wherein said session is included in a pool of sessions, said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job;
> launching said session to effect said execution of said job, if said session is available; and
> launching an error handling thread in response to an error condition, said error handling thread releasing said session.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re bond*, 910 F .2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. Kalman v. Kimberly-Clark Corp., 713 F .2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicants respectfully submit that Zolnowsky does not teach every element of the claimed invention arranged as they are in the claims. Specifically, Zolnowsky does not teach launching an error handling thread in response to an error condition, said error handling thread releasing said session.

Zolnowsky is directed to a process scheduler or dispatcher for a multiprocessor system for real time applications. The Zolnowsky system uses a dispatcher model that maintains a dispatch queue for each processor and a separate global dispatch queue for unbound higher priority real time threads. Each processor has its own queue and a dispatcher. Each queue has a separate schedule lock associated with it to protect scheduling operations. A processor's dispatcher selects a thread for execution from one of the queues in the system as a candidate thread to execute. When a candidate thread is selected for execution, the processor proceeds to verify against threads in the global real

time queue and the processor's own dispatch queue to select a highest priority runnable thread in the system.

Thus, the Zolnowsky system allows the dispatcher to prevent race conditions and minimize lock contention while assuring that high-priority threads are dispatched as quickly as possible. However, there is no section of the Zolnowsky reference that teaches launching an error handling thread in response to an error condition, said error handling thread releasing said session. With respect to original claim 5, the Office Action alleges that this feature is taught at column 8, lines 3-17, which reads as follows:

> The examination of the priorities on queues involve checking local variable associated with each processor and does not require any locks. For example, priority variable disp_maxrunpri can be used to indicate maximum priority level on a queue. Then variable disp_maxrunpri can be checked on both the processor dispatch queue and the real time queue using some suitable synchronization algorithm such as Dekker's algorithm to prevent miscommunication. However, any other suitable synchronization algorithm can be used in alternate embodiments of the present invention. Since the priority variables to be examined are atomic variables that are maintained in each dispatch queue, any scheduling errors caused by selecting a wrong queue will be caught in the a verification step. However, a schedule lock is required to take a thread from a selected queue.

This section of Zolnowsky describes the examination of priorities within the various queues. Zolnowsky teaches using a priority variable to indicate a maximum priority level and making use of a suitable synchronization algorithm to prevent miscommunication between a dispatch queue and a real time queue. Zolnowsky further teaches that any errors between the queues are caught in a verification step. The verification step referred to in this section is clarified by Zolnowsky at column 9, lines 42-50, which reads as follows:

> Thus, at step 604, a verification is made as to whether the selected thread is a best possible selection. This requires going back to the high priority real time queue and its own queue, and checking them again to see if there is a higher priority thread newly placed in the high priority real time queue or its own queue. If the selected candidate thread has higher priority than any other thread in either queue, then the processor continues to step 605 to execute the selected candidate thread.

Thus, Zolnowsky teaches a verification of a best possible thread selection by reconfirming the selected threads priority from the other threads in queue. There is nothing in these sections, or any other section of Zolnowsky, that teaches launching an error handling thread in response to an error condition, said error handling thread releasing said session. Even if the verification step were to be considered an error handling thread, the session is not released but merely institutes a thread substitution.

Thus, Zolnowsky does not teach each and every feature of independent claims 1, 12 and 23 as is required under 35 U.S.C. § 102. At least by virtue of their dependency on independent claims 1, 12 and 23, the specific features of dependent claims 2, 3, 13, 14, 24 and 25 are not taught by Zolnowsky. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1-3, 12-14 and 23-25 under 35 U.S.C. § 102.

Furthermore, Zolnowsky does not teach, suggest or give any incentive to make the needed changes to reach the presently claimed invention. Absent the Examiner pointing out some teaching or incentive to implement Zolnowsky such that an error handling thread is launched in response to an error condition and the error handling thread releases the session, one of ordinary skill in the art would not be led to modify Zolnowsky to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion or incentive to modify Zolnowsky in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the Applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

## III. 35 U.S.C. § 103, Alleged Obviousness, Claims 4, 6-9, 15, 17-20, 26 and 28-31

The Office Action rejects claims 4, 6-9, 15, 17-20, 26 and 28-31 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Zolnowsky (U.S. Patent No. 6,779,182 B1) in view of Northrup (U.S. Patent No. 6,671,713 B2). This rejection is respectfully traversed.

Claims 4, 6-9, 15, 17-20, 26 and 28-31 are dependent on independent claims 1, 12 and 23 and, thus, these claims distinguish over Zolnowsky for at least the reasons noted above with regards to claims 1, 12 and 23. Moreover, Northrup does not provide for the

deficiencies of Zolnowsky and, thus, any alleged combination of Zolnowsky and Northrup would not be sufficient to reject independent claims 1, 12 and 23 or claims 4, 6-9, 15, 17-20, 26 and 28-31 by virtue of their dependency. That is, Northrop does not teach launching an error handling thread in response to an error condition, said error handling thread releasing said session.

Moreover, in addition to their dependency from independent claims 1, 12 and 23, the specific features recited in dependent claims 4, 6-9, 15, 17-20, 26 and 28-31 are not taught or suggested by Zolnowsky and Northrup, either alone or in combination. With regard to claims 4, 15 and 26, the Office Action admits that Zolnowsky does not explicitly disclose determining if said connection is an existing connection, and wherein said step of creating said connection is performed if said connection is not an existing connection. However, the Office Action alleges that Northrup teaches this feature at column 4, lines 22-60, which reads as follows:

> The communication primitives are built using the underlying computer operating system intraprocess and interprocess communication facilities and thus are operating-system-specific. On one operating system there may be, for example, five communication primitives supported, while another operating system may support twenty. A communication primitive generally must provide for several operations to be applied such as:
>
> Create: The ability to create an instance of the primitive
> Destroy: The ability to destroy an instance of the primitive
> Send: The ability to send data to the primitive
> Receive: The ability to receive data from the primitive
> Cycle: Send a default and receive default messages to/from the primitive
> Connect: Primitive specific connection function
> Disconnect: Primitive specific disconnection function
> Suspend: Primitive specific suspension function
> Resume: Primitive specific resumption function
>
> Communication primitives are registered with the Thread Communication Service for the specific operating system the TCS is executing on. The name, the location, and certain characteristics describing the communication primitive are retained by the TCS for subsequent use. In this context, the communication primitives become a reusable asset, needing to be developed and tested only one time.
>
> Each communication primitive has a shared object, referred to as the communication primitive object, describing the location of the various operations to be applied when using this primitive type. All primitives have the same communication primitive object structure. The TCS will

load the communication primitive object at runtime only when requested for use by a communication point.

In a sense, the communication primitive can be thought of as analogous to the physical connection of a telephone on a phone network. A twisted pair telephone would use one primitive while a cellular telephone would use a different primitive.

All this section of Northrup teaches is communication primitives that are the low-level mechanisms used to provide the physical communication between various processes. The processes participating in the communication are referred to as communication points. Two or more communication points are connected by a communication link using the communication primitives. There is nothing in this section, or any other section of Northrup, that teaches or suggests determining if a connection is an existing connection, and, if the connection is not an existing connection, creating a connection. Thus, Zolnowsky and Northrup, either alone or in combination, fail to teach the features recited in claims 4, 15 and 26.

With regard to claims 6, 17 and 28, the Office Action admits that Zolnowsky does not explicitly disclose changing value of a job state from a first value to a second value in response to said launching of said error handling thread. However, the Office Action alleges that Northrup teaches this feature at column 56, lines 33-36, column 55, lines 27-35 and column 27, line 66 to column 28, line 15, which read as follows:

> Finally, the State Machine Thread Manager provides a special "Error" state to which a datum will transition when the exiting value of the State Thread is undefined in the current state machine.

(Column 56, lines 33-36)

> Using a standard Application Programming Interface, the Application Process can read the standard output of the logically named NEE to which it is ATTACHED. This is accomplished by the Minor Service Reader/Writer Thread of the NEEM that is reading the actual standard output of the NEE. It will send the standard output to the Application Process. Likewise, the Minor Service Reader/Writer Thread of the NEEM which reads the actual standard error will send the standard error output to the Application Process.

(Column 55, lines 27-35)

> When a new Service Thread is made available to the computer system, it can register its service by calling the Thread Directory Service

> specifying a REGISTER operation and providing the required information along with any optional information or attributes. Alternatively, a separate application can register other Service Threads available to the computer system by calling the Thread Directory Service and specifying a REGISTER operation along with the appropriate information. This permits a separate application program to provide this information without requiring the Service Thread to register itself. Duplicate entries in a given Thread Service Directory are not permitted. In this instance, the Thread Directory Service will return an error indication to the registering thread. In registering the Service Thread, the Thread Directory Service will assign a unique Thread Communication Identifier to be associated with the Service Thread. The Thread Directory Service will then return this identifier to the thread registering the service.

(Column 27, line 66 to column 28, line 15)

In column 55, lines 33-36, Northrup describes providing an error state to which a datum will transition when the exiting value of the state thread is undefined in the current state machine. However, the state change is not in response to the launching of an error handling thread. In column 55, lines 27-35, Northrup describes using a standard Application Programming Interface (API) to read the output of the logically Named Execution Environment to which the API is attached, which may include reading a standard error. All this section teaches is reading an error and does not teach or suggest the launching of an error handling thread. In column 27, line 66 to column 28, line 15, Northrup describes the addition of new service threads. None of these sections teaches or suggests changing value of a job state from a first value to a second value in response to said launching of said error handling thread. Thus, Zolnowsky and Northrup, either alone or in combination, fail to teach the features recited in claims 6, 17 and 28.

With regard to claims 7, 18 and 29, the Office Action alleges that Zolnowsky teaches said first value signals that said job is available for scheduling at column 8, lines 11-17, shown above. As discussed above, this section of Zolnowsky describes the examination of priorities within the various queues and the use of a priority variable to indicate a maximum priority level and making use of a suitable synchronization algorithm to prevent miscommunication between a dispatch queue and a real time queue. However, Zolnowsky further teaches that any errors between the queues are caught in a verification step and that the verification of a best possible thread selection by reconfirming the selected threads priority from the other threads in queue. There is

nothing in these sections, or any other section of Zolnowsky, that teaches a first value that signals that a job is available for scheduling. Thus, Zolnowsky and Northrup, either alone or in combination, fail to teach the features recited in claims 7, 18 and 29.

With regard to claims 8, 19 and 30, the Office Action alleges that Zolnowsky teaches retrying said steps of determining if a job is available for scheduling, determining if a session is available, and launching said session, in response to an error condition at column 8, lines 11-17, shown above. As discussed above, this section of Zolnowsky describes the verification of the priorities of threads and, if a higher priority thread exist, substituting the higher priority thread. Thus, Zolnowsky and Northrup, either alone or in combination, fail to teach the features recited in claims 8, 19 and 30.

With regard to claims 9, 20 and 31, the Office Action admits that Zolnowsky does not explicitly disclose retrying is repeated until a predetermined time interval has elapsed. However, the Office Action alleges that the use and advantages for retrying tasks based upon elapsed time is well known to one skilled in the art at the time the invention was made as evidence by the teachings of Northrup at column 10, line 49 to column 11, line 18, which reads as follows:

> The Application Process uses the Configuration Administrator Minor Service to administer zero or more components of software from shared libraries. Each component is said to offer a Minor Service. The specifications for the administration of the Minor Services can be provided directly by the Application Service, or, indirectly through a data store monitored by the Configuration Administrator. These specifications can instruct the Configuration Administrator Minor Service to perform the desired operation immediately, at a predefined time (which may be an interval), or, as a result of some event which is later communicated to the Configuration Administrator Minor Service.
>
> The Configuration Administrator Minor Service provides the following operations:
>
> 1. Locates specified Minor Services
> 2. Loads specified Minor Services
> 3. Executes specified Minor Services
> 4. Establishes communication channel with the specified Minor Service.
> 5. Suspends execution of specified Minor Services
> 6. Resumes execution of specified Minor Services
> 7. Replaces specified Minor Service with a new Minor Service rerouting communication channels as appropriate
> 8. Unloads specified Minor Service

9. Provides for manual state retention between replaceable Minor Services
10. Notification

Note that the Configuration Administrator Minor Service operations can be specified to occur at set time intervals; at predefined time periods; as a result of external events; or, as a result of internal events. Events, in this context are registered with the Configuration Administrator Minor Service to denote their occurrence.

While this section of Northrup may teach performing desired operations at a predefined time, this section does not teach or suggest repeating retrying the steps of determining if a job is available for scheduling, determining if a session is available, and launching a session, in response to an error condition, until a predetermined time interval has elapsed. Applicants respectfully submit that it would not be obvious to one skilled in the art at the time the invention was made to retry tasks based upon elapsed time in response to an error condition, as neither of the applied references teach or suggest this feature. Thus, Zolnowsky and Northrup, either alone or in combination, fail to teach the features recited in claims 9, 20 and 31.

Moreover, the Office Action may not use the claimed invention as an "instruction manual" or "template" to piece together the teachings of the prior art so that the invention is rendered obvious. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Such reliance is an impermissible use of hindsight with the benefit of Applicants' disclosure. *Id.* Therefore, absent some teaching, suggestion, or incentive in the prior art, Zolnowsky and Northrup cannot be properly combined to form the claimed invention. As a result, absent any teaching, suggestion, or incentive from the prior art to make the proposed combination, the presently claimed invention can be reached only through an impermissible use of hindsight with the benefit of Applicants' disclosure a model for the needed changes.

In view of the above, Zolnowsky and Northrup, taken either alone or in combination, fail to teach or suggest the specific features recited in dependent claims 4, 6-9, 15, 17-20, 26 and 28-31. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 4, 6-9, 15, 17-20, 26 and 28-31 under 35 U.S.C. § 103.

## IV. 35 U.S.C. § 103, Alleged Obviousness, Claims 10, 11, 21, 22, 32 and 33

The Office Action rejects claims 10, 11, 21, 22, 32 and 33 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Zolnowsky (U.S. Patent No. 6,779,182 B1) in view of Northrup (U.S. Patent No. 6,671,713 B2) and further in review of Rangarajan et al. (U.S. Patent No. 6,260,077 B1). This rejection is respectfully traversed.

Claims 10, 11, 21, 22, 32 and 33 are dependent on independent claims 1, 12 and 23 and, thus, these claims distinguish over Zolnowsky for at least the reasons noted above with regards to claims 1, 12 and 23. Moreover, Northrup and Rangarajan do not provide for the deficiencies of Zolnowsky and, thus, any alleged combination of Zolnowsky, Northrup and Rangarajan would not be sufficient to reject independent claims 1, 12 and 23 or claims 10, 11, 21, 22, 32 and 33 by virtue of their dependency.

In view of the above, Zolnowsky, Northrup and Rangarajan, taken either alone or in combination, fail to teach or suggest the specific features recited in independent claims 1, 12 and 23, from which claims 10, 11, 21, 22, 32 and 33 depend. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 10, 11, 21, 22, 32 and 33 under 35 U.S.C. § 103.

## V. Conclusion

It is respectfully urged that the subject application is patentable over the prior art of record and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: November 29, 2004

Francis Lammes
Reg. No. 55,353
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicants